SNAP - SN Application Proxy

Standard Performance Testing with MPI Only and MPI+OpenMP

Contacts
Al McPherson (LANL) – mcpherson@lanl.gov
Jonathan Robey (LANL) – jrobey@lanl.gov
Joe Zerr (LANL) – rzerr@lanl.gov
Randy Baker (LANL) – rsb@lanl.gov

SNAP Resources
Email: snap@lanl.gov
Website: https://github.com/losalamos/snap

Purpose

These test problems are meant to gauge system performance with problems typically encountered in the discrete ordinates transport community. This benchmark stresses the memory subsystem and total memory capacity. It also has the ability to use newer MPI and OpenMP features, such as nested threads and thread multiple communication when available.

Characteristics

The discrete ordinates transport equation is notorious for memory consumption. A spatially 3-D, time-dependent calculation using the discrete ordinates proxy app, SNAP, requires the storage of two six-dimensional arrays whose sizes each equal the number of spatial cells by the number of particle energy bins by the number of discrete ordinates to angularly resolve the problem. SNAP uses a parallel model that decomposes the spatial dimension over a distributed memory system via MPI. Work performed in loops over energy groups is threaded. And angles of a given octant are vectorized using the vector instructions sets on chips.

Typically, transport calculations are designed with the competing goals of properly resolving the physics of the problem while also ensuring that the needed data structures will fit within the available memory. Included in this document is a table with a sample set of weak scaling problems that were used to derive the figure of merit (FOM) and scaling for SNAP. The problem has a number of angles and energy bins, “groups,” typical of current application space. Twice the hardware should be used to increase the number of spatial cells between each problem. The actual spatial bounds are fixed, and more cells yields finer spatial resolution.

These problems have been sized for a system that assumes 1 GB memory per MPI task.

Each of the examples used about 0.5 GB per MPI task. Importantly, the problems being run should take up only approximately half the available memory to be realistic and representative.

Mechanics of Building Benchmark

For a simple build edit the Makefile to the appropriate compiler and flags for your system. Then type 'make' to build.

For a more complex build the code has various advanced features commented out, such as nested threads and thread multiple communication. Add these back in as desired.

Mechanics of Running Benchmark

Use the appropriate command line options to initiate the MPI application, set the number of processes, set the number of threads, point at the SNAP executable, and define the input and output file names. See the table below for guidance on changing the number of cells with increasing core count (the highlighted row indicates the problem used for the baseline FOM).

1. Small problem: use the single core problem from the MPI-only test suite.
2. Medium problem: use the MPI-only tests and change the "nthreads" variable as desired to add in threads for a parallel calculation using hundreds of processes.
3. Large CORAL reference (FOM baseline) problem: follow the spreadsheet for weak scaling problems and add in threads as desired.
4. CORAL problem: modify the 65536 MPI rank problem to increase total problem size by a factor of two. The suggested modification is to increase group count to 80, but other options, such as doubling the number of angles or doubling "nz" will have the same result on problem size. To increase the amount of strong scaling, either up the number of MPI ranks or number of threads as appropriate for your system.

For all calculations, please ensure the input "nx" is evenly divisible by the input "ichunk," "ny" by "npey," and "nz" by "npez." SNAP will abort if these rules are not met.

It is suggested you try multiple "ichunks" to find the one that performs best on your machine. This is the parameter that impacts performance the greatest as its changed.

The reported "Grind Time" is a useful measure of SNAP performance. It is the time to compute the solution for one phase space cell (space, angle, and energy) in a single iteration. It will decrease linearly in the ideal case, but deviate from the ideal due to communication penalties in both strong and weak scaling cases.

MPI Ranks/ Cores	nx	ny	nz	Total cells	ichunk	npey	npez	nang	ng	nthreads	Total Threads
16	32	32	32	32768	8	4	4	50	40	4	64
32	32	32	64	65536	8	4	8	50	40	4	128
64	32	64	64	131072	8	8	8	50	40	4	256
128	64	64	64	262144	8	8	16	50	40	4	512
256	64	64	128	524288	8	16	16	50	40	4	1024
512	64	128	128	1048576	8	16	32	50	40	4	2048
1024	128	128	128	2097152	8	32	32	50	40	4	4096
2048	128	128	256	4194304	8	32	64	50	40	4	8192
4096	128	256	256	8388608	8	64	64	50	40	4	16384
8192	256	256	256	16777216	8	64	128	50	40	4	32768
16384	256	256	512	33554432	8	128	128	50	40	4	65536
32768	256	512	512	67108864	8	128	256	50	40	4	131072
65536	512	512	512	134217728	8	256	256	50	40	4	262144
131072	512	512	1024	268435456	8	256	512	50	40	4	524288

Table 1 - Input parameters used for baseline runs of different sizes

```
input template
&invar
  nthreads=4
  Nnested=0
  npey=#
  npez=#
  ichunk=#
  ndimen=3
  nx=#
  Lx=4.8
  ny=#
  Ly=4.8
  nz=#
  Lz=4.8
  Nmom=3
  Nang=50
  Ng=400
  mat_opt=1
  src_opt=0
  timedep=1
  it_det=0
  Tf=0.04
```

```
  Nsteps=4
  iitm=5
  oitm=100
  epsi=1.E-4
  fluxp=0
  scatp=0
  Fixup=1
! soloutp=0 \
```

Verification of Results

SNAP has been designed to mimic the workflow and communication patterns of the LANL transport code PARTISN. However, to ensure approval for open source status, the operators have been modified, meaning the same problem run with the two codes will not produce the same result. Moreover, some numerical instability may appear with SNAP that will not appear in PARTISN.

Fortunately, steps can be taken to create a base set of reference cases for SNAP that one can use to ensure their base copy built correctly or, perhaps more valuably, that a modified SNAP has not altered the solution algorithms. The "qasnap" directory within the GitHub repository provides some simple regression tests using up to 16 cores. There are two main goals of this approach.

1. Because the solution is unphysical and unrealistic to begin with, one should attempt to use base SNAP code for verifying results of a modified SNAP.
2. Reference SNAP calculations should converge and provide a result that is easily compared to other calculations with a modified SNAP. Non-converging results can still be useful for timing measurements, but are not valuable for ensuring any modifications have not affected the solution algorithms.

No single method is available to ensure these benchmark problems have run correctly. Yet, the following steps are recommended to verify results. First, the benchmark problems have been designed with the expectation that the solution will converge successfully, providing useful output for review. Second, this solution should be symmetric given the symmetric source and material layout. Users may view the flux solution in the output to ensure that it is in fact symmetric. Users also should compare the number of transport mesh sweeps, "inners," performed for a calculation for further verification that a modified SNAP performs the same as a base SNAP calculation. The number of inners may serve as a potential figure of merit, assuming nothing about the sweep procedure has changed. That is, the decomposition of work may change, but the order of operations should remain the same, and the number of operations to reach convergence has also not changed. These steps should help the user feel confident that SNAP is computing the same solution as the base SNAP code.

The figure of merit (FOM) is calculated by taking the inverse from the grind time. Both solving a twice as big problem in the same amount of time and solving the same size problem in half the time will double the FOM.

Any difficulties with SNAP execution or questions about the solution should be directed to snap@lanl.gov.